MASTER DAPM - Projet M1, année 2014-2015

Vote électronique

Table des matières

Enseignants référents

- Emmanuel Bruno
- Philippe Langevin
- Elisabeth Murisasco
- Christian Nguyen
- Pascal Véron
- Jean-Pierre Zanotti

Organisation du projet

Les étudiants sont répartis en 5 groupes de 7 étudiants. Des revues d'avancement de 30mn pour chaque groupe sont prévues environ tous les 10 jours. Les groupes seront constitués par tirage au sort lors de la présentation du projet.

Le sujet du projet sera fourni lors d'une présentation le **lundi 20 avril**. Le projet durera environ 7 semaines.

Les étudiants travaillent sur le projet les semaines suivantes:

- du 20 avril
- vacances semaine du 27 avril
- du 4 mai
- du 11 mai
- du 18 mai
- du 26 mai
- du 1er juin
- du 8 juin : soutenance le vendredi 12 juin + jury M1 dans la foulée

Les étudiants sont libérés de tout autre module. Il s'agit de réaliser - ***à temps plein*** - un projet en groupe qui s'appuie, en particulier, sur les modules de l'année. Les salles **U026** et **U'110** sont mises à la disposition des étudiants.

L'avancement du travail de chaque groupe est présenté lors de réunions régulières organisées avec le jury d'enseignants. Lors de ces réunions, chaque groupe présente (à l'aide de 3-4 supports) avec 20mn de présentation et 10mn de discussion:

- la répartition du travail au sein de l'équipe
- les avancées dans la démarche, la formalisation, l'implantation
- les difficultés rencontrées
- le travail de la semaine à venir
- des perspectives pour les semaines suivantes

Tous les étudiants du groupe devront parler lors de ces réunions. À charge des groupes d'équilibrer l'organisation de ce tour de parole sur l'ensemble des réunions.

Rendu

A l'issue du projet, chaque groupe devra fournir :

- une application fonctionnelle, le code source documenté et un mode d'emploi (sous la forme d'un site Web).
- des supports pour la présentation orale (avec un temps de parole partagé pour les membres du groupe)
- une démonstration
- des éléments pour apprécier le travail en équipe (cf. ci-dessous "Travail en équipe")

Planning

- présentation du sujet : lundi 20 avril 14h
- présentation d'algorithme de votes : lundi 20 avril 15h
- SPRINT 0 au minimum jusqu'à la première réunion
- réunions régulières (20 mn par groupe + 10 mn d'échange) :
    * vendredi 24 avril 9h
    * lundi 11 mai 14h
    * mardi 19 mai 14h

* mardi 26 mai 9h
* jeudi 4 juin 9h

- présentation des projets/soutenance : 12 juin 9h.

Travail en équipe

Pour organiser le travail en équipe, il est demandé à chaque groupe de suivre une méthodologie dite *agile* (présentée dans le module D12 avec une introduction à *Scrum* pour la gestion d'un projet). *Scrum* propose un découpage des fonctionnalités du projet en itérations courtes (les sprints) ainsi qu'une auto-organisation de l'équipe de développement. Nous vous proposons de suivre une organisation présentée ici SCRUM que vous pourrez adapter. Dans tous les cas, le jury devra disposer au moins lors des réunions d'éléments concrets d'appréciations. Pour plus de détails : "*Scrum le guide pratique de la méthode agile la plus populaire"* Claude Aubry, Dunod.

Chaque équipe disposera d'un espace sur REDMINE-EDU pour gérer son projet (suivi de l'avancement et partage du code source "basique"[1] avec Git). Le nom du projet redmine sera de la forme : ProjetM1_20142015_G*i* avec *i* le numéro du groupe. **L'utilisation de Git sur cet outil est obligatoire.** Reportez-vous à GIT-MINIMAL pour une introduction rapide à Git.

Objectif général

L'objectif du projet est de définir et de réaliser un système de gestion (création et consultation) de votes électronique. Les votes et leur administration se feront principalement depuis une application mobile androïd et éventuellement depuis une application web. **Un soin particulier sera apporté pour proposer un système qui exploite la particularité d'être une application mobile (interface adaptée, utilisation des capteurs, spécificité de la mobilité, ...).**

Éléments de réflexion

Cette partie présente ce que peut être un système de gestion de vote. Différents aspects sont présentés, tous ne doivent pas forcément être implantés, il s'agit d'une base pour la réflexion.

Vote "simple"

- Voting system

Un système de vote (*voting system*) est une méthode qui permet à des votants (*voter*) de

[1] Basique : *add, commit, push* et *pull* en évitant les conflits en travaillant sur des fichiers séparés.

choisir parmi une ou plusieurs options (*option*). Le système de vote vérifie que les règles de vote sont respectées, applique un algorithme de comptage de votes et agrège les résultats pour fournir le résultat final (*tallying method*).

Les règles de vote définissent les contraintes éventuelles du vote : minimum du nombre de votes exprimés (*quorum*), gestion des procurations (*proxy voting*), début et fin du scrutin (manuelle, date/durée, nombre de votant, ...), modification autorisée, contraintes de consultation (qui: certains utilisateurs ou groupes, quoi: résultats agrégés, détails, ... et quand: toujours, après avoir répondu, jamais, ...), ...

Les options sont présentées aux utilisateurs dans un bulletin de vote (*ballot*). Les utilisateurs peuvent exprimer leur choix de différentes façons :

- par classement des options : une liste ordonnée d'options, éventuellement partielle (voire limitée dans les contraintes de vote) et avec éventuellement des ex-aequo.
- par approbation : pour chaque option chaque votant indique oui/non/sans avis.
- par notation : chaque option est notée (dans un intervalle) par chaque votant.
- ...

Le résultat du vote peut être une option, un ensemble ou une liste d'options en fonction du mode d'expression des préférences et de l'algorithme d'agrégation.

Le vote de chaque utilisateur peut éventuellement être pondéré (nombre de parts dans une organisation, place dans la hiérarchie, ...).

Vote "complexe"

Un système de vote peut être plus complexe en proposant plusieurs tours ayant éventuellement des règles différentes (élimination, majorité relative/absolue, quorum, ...). Le calcul du résultat peut aussi se faire avec un autre algorithme en fonction du résultat du vote : par exemple pour un vote dans lequel chaque option est une liste bloquée, la répartition dans le résultat final des éléments de chaque liste peut est faite avec un autre calcul : proportionnelle intégrale avec ou sans/seuil, proportionelle mixte (majorité absolue pour le vainqueur,... , à comparer avec le vote préférentiel).

Les algorithmes de vote

Il existe de très nombreux algorithmes de vote et différents critères pour les évaluer voire les comparer. A theory of measuring, electing, and ranking est une bonne introduction à la problématique. Evaluating voting systems using criteria, Arrow's Impossibility Theorem, Satterthwaite Theorem présentent les critères principaux.

Quelques pistes :
Vote majoritaire (un choix)

Votes préférentiels (classement):

- Borda : Classement par somme des préférences de chaque votant pour chaque option.
- Single Transferable Vote : Chaque bulletin est associé à la première option. S'il n'y a pas de vainqueur majoritaire, on élimine une option (plusieurs critères d'élimination possibles : plus de défaites, moins bon total, ...) puis on transfère les bulletins de l'option éliminée à l'option suivante sur celui-ci. On répète le processus jusqu'à l'obtention d'un vainqueur.
- Les méthodes de Condorcet : Basées sur des comparaisons par paires des candidats (X, Y) pour remplir une table (Préfère X sur Y, Égal, Préfère Y sur X). Les variantes règlent les situations dans lesquelles il n'y a pas de vainqueur :
  - Ranked Pair : les paires sont ordonnées.
  - Méthode black : Condorcet puis Borda si pas de gagnant.
  - Schulze http://link.springer.com/article/10.1007%2Fs00355-010-0475-4
  - Kemeny–Young method : Le résultat est la minimisation de la distance kendal-Tau entre les bulletins.

Votes par évaluation (notation): http://en.wikipedia.org/wiki/Cardinal_voting_systems

- Jugement majoritaire (SW) Le vainqueur est celui qui à la meilleure moyenne (meilleur si avis sensés être convergent ? ex. jury)
- Somme des notations.
- Vote par approbation (Version + simple sans classement mais avec un choix OUI/NON)

Quelques Comparaisons : System Comparison, Schulze method, Vote de valeur, Vote au pluriel.

Fonctionnalités attendues

- Un système de vote simple permettant de paramétrer et de réaliser un vote en ayant le choix d'un algorithme parmi plusieurs voire de le paramétrer. Une attention particulière sera apportée au minimum aux trois algorithmes suivants : Single Transferable Vote, Kemeny–Young Method et un vote par évaluation avec choix paramétrable (somme, moyenne, ...).
- Le code sera documenté de façon appropriée et associé à des tests unitaires, des tests d'intégration et des tests fonctionnels.
- Après un état de l'art sur les techniques de mesures des performances des application java et android. Des mesures de performances sur un jeu de données réalistes devront être fournies avec l'application.

Architectures attendues

1. Clients/serveur (Obligatoire). Un serveur de vote offrant : (i) une API REST permettant au moins d'administrer les votes et les utilisateurs, de consulter le détail d'un vote, de voter, d'obtenir le contenu de l'urne (anonymisé ou non), de recevoir

un résultat et de l'afficher. (ii) Une connexion websocket permettant de notifier (en fonction des paramètres) les clients des évènements liés au vote (ouverture, vote, fin, ...). **Le calcul du résultat sera effectué sur un ou plusieurs des clients et transmis au serveur**. L'api REST devra être particulièrement soignée (Un point de départ : Best Practices for Designing a Pragmatic RESTful API de Vinay Sahni et l'api de github). Dans cette version le serveur est une sorte de bus de messages.

2. Les aspects liés à la sécurités (authentification, confidentialité, ...) seront traités plus tard (des informations complémentaires seront fournies).
3. En fonction de l'avancement, une seconde version de l'application ne nécessitant pas de serveur pourra être étudiée et implantée (en remplaçant l'api REST et les WebSockets par WebRTC).